

Máster Avanzado de Programación en Python para Hacking, Big Data y Machine Learning

BUENAS PRÁCTICAS DE
PROGRAMACIÓN EN PYTHON

LECCIÓN 02

Comprender la importancia del desarrollo guiado por pruebas

ÍNDICE

Presentación y objetivos

Test unitarios

Test unitarios con pytest

Test unitarios con unittest

Limitaciones test unitarios

INTRODUCCIÓN

En esta lección aprenderemos qué es un test unitario, porqué utilizarlos en nuestras implementaciones y cuáles son sus principales ventajas e inconvenientes.

Además, aprenderemos a utilizar dos de las herramientas más comunes en Python para crear test unitarios. En concreto desarrollaremos un conjunto de tests para validar las diferentes funciones implementadas en nuestros programas.

OBJETIVOS

Al finalizar esta lección serás capaz de:

- 1 Comprender la importancia del desarrollo de código guiado por pruebas.
- 2 Comprender cuáles son las principales limitaciones de los test unitarios.
- 3 Conocer y utilizar las herramientas pytest y unittest para validar nuestro código.
- 4 Conocer las diferencias y limitaciones entre pytest y unittest.

Test unitarios

En programación, realizar un test unitario es el proceso por el cual podemos comprobar el correcto funcionamiento de un fragmento de la totalidad del código.

En términos generales, la idea consiste en escribir un conjunto de casos de prueba para cada función o método implementado para asegurar el correcto funcionamiento de cada módulo por separado.

El principal objetivo es asegurar que cada unidad funciona correctamente y que además, el código que ha sido implementado cumple con su cometido.

Test unitarios

Cuando implementamos un conjunto de pruebas unitarias sobre nuestro software, hemos de comprobar que se cumplan los siguientes requisitos:

- **Automatizable.** Una vez implementado el conjunto de test unitarios, estos deberían tener la capacidad de ser ejecutados y validados por sí mismos, sin necesidad de la intervención manual del programador.
- **Completas.** Dichos tests deben cubrir la mayor cantidad posible del código desarrollado.
- **Reutilizables.** Han de ser ejecutables más de una vez.
- **Independientes.** El funcionamiento de un test debe ser independiente del resto de pruebas unitarias.
- **Profesionales.** Deben ser desarrollados minuciosamente y con la misma profesionalidad con que ha sido desarrollado el resto del código.

Test unitarios con Pytest

Es una librería de Python diseñada para facilitar la tarea del desarrollador a crear test unitarios.

Utilizar test unitarios en tus desarrollos aumenta la confianza para con tus implementaciones, ya que se tiene la certeza de que el código se comporta como espera y asegura que los cambios en el código no causarán un daño mayor.

Con pytest, las tareas comunes requieren de menos código mientras que las tareas más avanzadas pueden ser realizadas con una amplia variedad de comandos y complementos que incluye pytest.

Test unitarios con Pytest

```
ramon@ramon-rd: ~/Escritorio/ev
(fp) (base) ramon@ramon-rd:~/Escritorio/ev$ pip install pytest
Collecting pytest
  Downloading pytest-6.2.3-py3-none-any.whl (280 kB)
    |████████████████████| 280 kB 2.1 MB/s
Requirement already satisfied: pluggy<1.0.0a1,>=0.12 in /home/ramon/miniconda3/lib/python3.8/site-packages (from pytest) (0.13.1)
Collecting iniconfig
  Using cached iniconfig-1.1.1-py2.py3-none-any.whl (5.0 kB)
Requirement already satisfied: packaging in /home/ramon/miniconda3/lib/python3.8/site-packages (from pytest) (20.9)
Collecting py>=1.8.2
  Using cached py-1.10.0-py2.py3-none-any.whl (97 kB)
Requirement already satisfied: toml in /home/ramon/.local/lib/python3.8/site-packages (from pytest) (0.10.2)
Requirement already satisfied: attrs>=19.2.0 in /home/ramon/miniconda3/lib/python3.8/site-packages (from pytest) (20.3.0)
Requirement already satisfied: pyparsing>=2.0.2 in /home/ramon/miniconda3/lib/python3.8/site-packages (from packaging->pytest) (2.4.7)
Installing collected packages: iniconfig, py, pytest
Successfully installed iniconfig-1.1.1 py-1.10.0 pytest-6.2.3
(fp) (base) ramon@ramon-rd:~/Escritorio/ev$
```

Test unitarios con Pytest

```
ramon@ramon-rd: ~/Escritorio/ev
(py) (base) ramon@ramon-rd:~/Escritorio/ev$ py.test -h
usage: py.test [options] [file_or_dir] [file_or_dir] [...]

positional arguments:
  file_or_dir

general:
  -k EXPRESSION          only run tests which match the given substring
                        expression. An expression is a python evaluable
                        expression where all names are substring-matched against
                        test names and their parent classes. Example: -k
                        'test_method or test_other' matches all test functions
                        and classes whose name contains 'test_method' or
                        'test_other', while -k 'not test_method' matches those
                        that don't contain 'test_method' in their names. -k 'not
                        test_method and not test_other' will eliminate the
                        matches. Additionally keywords are matched to classes
                        and functions containing extra names in their
                        'extra_keyword_matches' set, as well as functions which
                        have names assigned directly to them. The matching is
                        case-insensitive.
  -m MARKEXPR           only run tests matching given mark expression.
                        For example: -m 'mark1 and not mark2'.
  --markers             show markers (builtin, plugin and per-project ones).
  -x, --exitfirst       exit instantly on first error or failed test.
  --fixtures, --funcargs show available fixtures, sorted by plugin appearance
                        (fixtures with leading '_' are only shown with '-v')
  --fixtures-per-test  show fixtures per test
  --pdb               start the interactive Python debugger on errors or
                        KeyboardInterrupt.
  --pdbcls=modulename:classname
                        start a custom interactive Python debugger on errors.
                        For example:
                        --pdbcls=IPython.terminal.interactiveshell:TerminalPDB
```

Test unitarios con Pytest

Caso práctico...

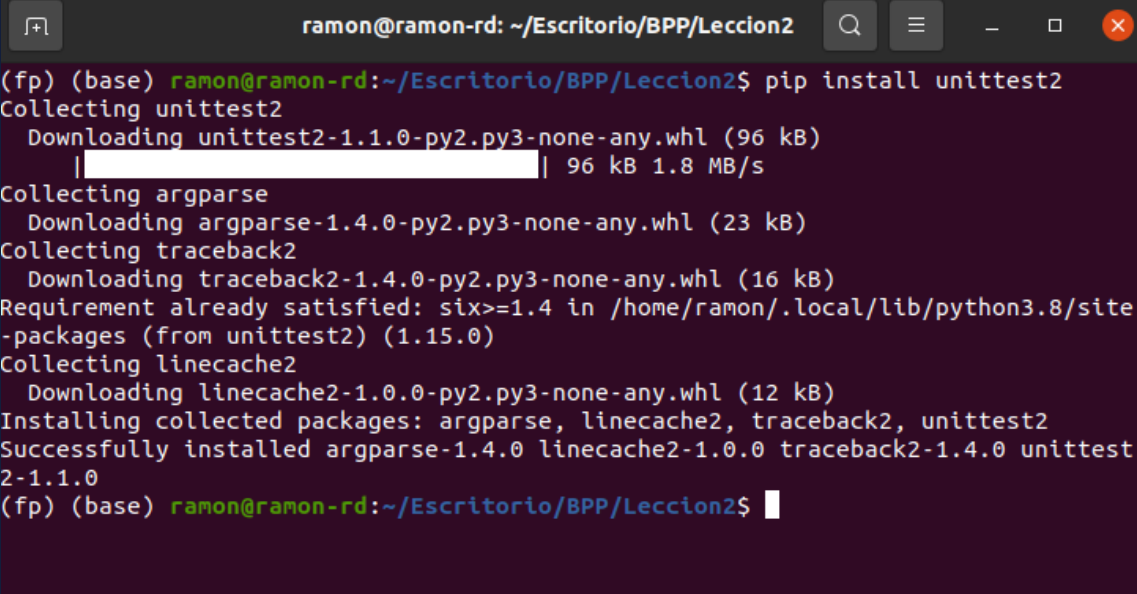
Test unitarios con Unittest

Unittest es una alternativa a pytest para implementar test unitarios en Python. Esta herramienta es una de las más utilizadas en Python y permite realizar pruebas desde el propio código, heredando de la clase unittest.

Con unittest las pruebas pueden arrojar uno de los siguientes resultados:

- **OK.** Nos indica que la prueba ha sido ejecutada correctamente y que no se ha encontrado ningún problema.
- **FAIL.** Indica que la prueba no ha podido ser ejecutada y sería necesario incluir una excepción (try/except).
- **ERROR.** Indica que el fragmento de código no ha pasado el test.

Test unitarios con Unittest



```
ramon@ramon-rd: ~/Escritorio/BPP/Leccion2
(fp) (base) ramon@ramon-rd:~/Escritorio/BPP/Leccion2$ pip install unittest2
Collecting unittest2
  Downloading unittest2-1.1.0-py2.py3-none-any.whl (96 kB)
    |████████████████████| 96 kB 1.8 MB/s
Collecting argparse
  Downloading argparse-1.4.0-py2.py3-none-any.whl (23 kB)
Collecting traceback2
  Downloading traceback2-1.4.0-py2.py3-none-any.whl (16 kB)
Requirement already satisfied: six>=1.4 in /home/ramon/.local/lib/python3.8/site-packages (from unittest2) (1.15.0)
Collecting linecache2
  Downloading linecache2-1.0.0-py2.py3-none-any.whl (12 kB)
Installing collected packages: argparse, linecache2, traceback2, unittest2
Successfully installed argparse-1.4.0 linecache2-1.0.0 traceback2-1.4.0 unittest2-1.1.0
(fp) (base) ramon@ramon-rd:~/Escritorio/BPP/Leccion2$
```

Test unitarios con Unittest

Caso práctico...

Test unitarios con Unittest

Method	Checks that
<code>assertEqual(a, b)</code>	<code>a == b</code>
<code>assertNotEqual(a, b)</code>	<code>a != b</code>
<code>assertTrue(x)</code>	<code>bool(x)</code> is True
<code>assertFalse(x)</code>	<code>bool(x)</code> is False
<code>assertIs(a, b)</code>	<code>a is b</code>
<code>assertIsNot(a, b)</code>	<code>a is not b</code>
<code>assertIsNone(x)</code>	<code>x is None</code>
<code>assertIsNotNone(x)</code>	<code>x is not None</code>
<code>assertIn(a, b)</code>	<code>a in b</code>
<code>assertNotIn(a, b)</code>	<code>a not in b</code>
<code>assertIsInstance(a, b)</code>	<code>isinstance(a, b)</code>
<code>assertNotIsInstance(a, b)</code>	<code>not isinstance(a, b)</code>

Test unitarios con Unittest

Method	Checks that
<code>assertAlmostEqual(a, b)</code>	<code>round(a-b, 7) == 0</code>
<code>assertNotAlmostEqual(a, b)</code>	<code>round(a-b, 7) != 0</code>
<code>assertGreater(a, b)</code>	<code>a > b</code>
<code>assertGreaterEqual(a, b)</code>	<code>a >= b</code>
<code>assertLess(a, b)</code>	<code>a < b</code>
<code>assertLessEqual(a, b)</code>	<code>a <= b</code>
<code>assertRegex(s, r)</code>	<code>r.search(s)</code>
<code>assertNotRegex(s, r)</code>	<code>not r.search(s)</code>
<code>assertCountEqual(a, b)</code>	<code>a</code> and <code>b</code> have the same elements in the same number, regardless of their order.

Test unitarios con Unittest

Method	Used to compare
<code>assertMultiLineEqual(a, b)</code>	strings
<code>assertSequenceEqual(a, b)</code>	sequences
<code>assertListEqual(a, b)</code>	lists
<code>assertTupleEqual(a, b)</code>	tuples
<code>assertSetEqual(a, b)</code>	sets or frozensets
<code>assertDictEqual(a, b)</code>	dicts

Limitaciones test unitarios

Mostraremos algunas de las limitaciones de los test unitarios, para que el estudiante comprenda a un mayor nivel de detalle cómo funcionan dichos test y cuando estos no pueden ser utilizados.

Debido a su facilidad de uso, en este caso práctico haré uso de la herramienta pytest.



CONCLUSIONES

1

Qué son los test unitarios y porque es importante incluirlos en nuestros desarrollos.

2

Utilizar la herramienta **pytest** para validar los diferentes métodos implementados en nuestro código.

3

Utilizar la herramienta **unittest** para validar los diferentes métodos implementados en nuestro código.

MUCHAS GRACIAS POR SU ATENCIÓN



rrueda@grupomainjobs.com



Ramón Rueda Delgado
<https://www.linkedin.com/in/ramon-rueda/>



twitter.com/eiposgrados



facebook.com/eiposgrados



instagram.com/eiposgrados